In the Claims:

1. (original) A method for mapping managed object metadata, the method comprising:

inputting a first data type from a first set of data types, wherein the first set of data types is expressed in an abstract syntax notation, and wherein the abstract syntax notation comprises a language for describing data;

determining a corresponding second data type from a second set of data types, wherein the second set of data types is expressed in an interface definition language, wherein the interface definition language comprises a language for implementing interfaces to managed objects, wherein the interface definition language is operable across a plurality of platforms and across a plurality of programming languages, and wherein the interface definition language is class independent; and

returning the second data type.

2. (original) The method of claim 1, wherein the managed object metadata comprises metadata concerning a telephone system.

3. (original) The method of claim 1, wherein the managed object metadata comprises metadata concerning a network switch.

4. (original) The method of claim 1, wherein the metadata comprises type information about an attribute of one of the managed objects.

5. (original) The method of claim 1, wherein the metadata comprises type information about an action of one of the managed objects.

6. (original) The method of claim 1, wherein the metadata comprises type information about a notification of one of the managed objects.

7. (original) The method of claim 1, wherein the first data type comprises a primitive data type in the abstract syntax notation, and wherein the second data type comprises a generic primitive data type in the interface definition language.

8. (currently amended) The method of claim 1, wherein the first data type comprises an object data type in the abstract syntax notation, and wherein the second data type comprises a sequence of a ~~the~~ generic primitive data type in the interface definition language.

9. (original) The method of claim 1, wherein the first data type comprises an object data type in the abstract syntax notation, wherein the second data type comprises a choice structure of the interface definition language, and wherein the choice structure comprises:

a generic value field;

a plurality of data types; and

a selector, wherein the selector is an index whose value determines which of the plurality of data types is used to interpret the value in the generic value field.

10. (original) The method of claim 1, wherein the abstract syntax notation comprises Abstract Syntax Notation One.

11. (original) The method of claim 1, wherein the abstract syntax notation comprises Guidelines for Managed Objects (GDMO).

12. (currently amended) The method of claim 1, wherein the interface definition language is operable to provide a ~~single~~ mapping which is applicable to any managed object class.

13. (currently amended) A method for mapping managed object metadata, the method comprising:

inputting a first data type from a first set of data types, wherein the first set of data types is expressed in an interface definition language, wherein the interface definition language comprises a language for implementing interfaces to managed objects, wherein the interface definition language is operable across a plurality of platforms and across a plurality of programming languages, and wherein the interface definition language is operable to provide a ~~single~~ mapping which is applicable to any managed object class;

determining a corresponding second data type from a second set of data types, wherein the second set of data types is expressed in an abstract syntax notation, and wherein the abstract syntax notation comprises a language for describing data; and

returning the second data type.

14. (original) The method of claim 13, wherein the managed object metadata comprises metadata concerning a telephone system.

15. (original) The method of claim 13, wherein the managed object metadata comprises metadata concerning a network switch.

16. (original) The method of claim 13, wherein the metadata comprises type information about an attribute of one of the managed objects.

17. (original) The method of claim 13, wherein the metadata comprises type information about an action of one of the managed objects.

18. (original) The method of claim 13, wherein the metadata comprises type information about a notification of one of the managed objects.

19. (original) The method of claim 13, wherein the first data type comprises a generic primitive data type in the interface definition language, and wherein the second data type comprises a primitive data type in the abstract syntax notation.

20. (currently amended) The method of claim 13, wherein the first data type comprises a sequence of a ~~the~~ generic primitive data type in the interface definition language, and wherein the second data type comprises an object data type in the abstract syntax notation.

21. (original) The method of claim 13, wherein the first data type comprises a choice structure of the interface definition language, wherein the choice structure comprises:

a generic value field;

a plurality of data types; and

a selector, wherein the selector is an index whose value determines which of the plurality of data types is used to interpret the value in the generic value field; and

wherein the second data type comprises an object data type in the abstract syntax notation.

22. (original) The method of claim 13, wherein the abstract syntax notation comprises Abstract Syntax Notation One.

23. (original) The method of claim 13, wherein the abstract syntax notation comprises Guidelines for Managed Objects (GDMO).

24. (currently amended) The method of claim 13, wherein the interface definition language is operable to provide a ~~single~~ mapping which is applicable to any managed object class.

25. (original) A computer system for mapping managed object metadata, wherein the system comprises:

a CPU;

a memory coupled to the CPU, wherein the memory stores program instructions executable by the CPU, and wherein the program instructions are executable to:

input a first data type from a first set of data types, wherein the first set of data types is expressed in an abstract syntax notation, and wherein the abstract syntax notation comprises a language for describing object data;

determine a corresponding second data type from a second set of data types, wherein the second set of data types is expressed in an interface definition language, wherein the interface definition language comprises a language for implementing interfaces to managed objects, wherein the interface definition language is operable across a plurality of platforms and across a plurality of

programming languages, and wherein the data types in the interface definition language are class independent; and

return the second data type.

26. (original) The computer system of claim 25, wherein the managed object metadata comprises metadata concerning a telephone system.

27. (original) The computer system of claim 25, wherein the managed object metadata comprises metadata concerning a network switch.

28. (original) The computer system of claim 25, wherein the abstract syntax notation comprises Abstract Syntax Notation One.

29. (original) The computer system of claim 25, wherein the abstract syntax notation comprises Guidelines for Managed Objects (GDMO).

30. (original) The computer system of claim 25, wherein in determining the corresponding second data type from the second set of data types, the program instructions are executable to look up the second data type in one or more lookup tables.

31. (original) A carrier medium comprising program instructions for mapping managed object metadata, wherein the program instructions are computer-executable to implement:

inputting a first data type from a first set of data types, wherein the first set of data types is expressed in an abstract syntax notation, and wherein the abstract syntax notation comprises a language for describing data;

determining a corresponding second data type from a second set of data types, wherein the second set of data types is expressed in an interface definition

language, wherein the interface definition language comprises a language for implementing interfaces to managed objects, wherein the interface definition language is operable across a plurality of platforms and across a plurality of programming languages, and wherein the interface definition language is class independent; and

returning the second data type.

32. (currently amended) The carrier medium of claim 31, wherein the first data type comprises an object data type in the abstract syntax notation, and wherein the second data type comprises a sequence of a ~~the~~ generic primitive data type in the interface definition language.

33. (currently amended) A carrier medium comprising program instructions for mapping managed object metadata, wherein the program instructions are computer-executable to implement:

inputting a first data type from a first set of data types, wherein the first set of data types is expressed in an interface definition language, wherein the interface definition language comprises a language for implementing interfaces to managed objects, wherein the interface definition language is operable across a plurality of platforms and across a plurality of programming languages, and wherein the interface definition language is operable to provide a ~~single~~ mapping which is applicable to any managed object class;

determining a corresponding second data type from a second set of data types, wherein the second set of data types is expressed in an abstract syntax notation, and wherein the abstract syntax notation comprises a language for describing data; and

returning the second data type.

34. (currently amended) The carrier medium of claim 33, wherein the first data type comprises a sequence of a ~~the~~ generic primitive data type in the interface definition language, and wherein the second data type comprises an object data type in the abstract syntax notation.